Analysis Components Investigation Report

Sébastien Paquet
Fujitsu Consulting Canada Inc.

Prepared By:
Fujitsu Consulting Canada Inc.
2000 Boulevard Lebourgneuf,
Bureau 300, Québec (Québec) G2K 0B8
PWGSC Contract Number: W7701-13-5551
CSA: Alexandre Bergeron-Guyard, DRDC Valcartier, Defence Scientist, 418-844-4000 Ext. 4107

The scientific or technical validity of this Contract Report is entirely the responsibility of the Contractor and the contents do not necessarily have the approval or endorsement of Department of National Defence of Canada.

The information contained herein is proprietary to Her Majesty and is provided to the recipient on the understanding that it will be used for information and evaluation purposes only. Any commercial use including use for manufacture is prohibited.

Defence Research and Development Canada

Contractor Report
DRDC-RDDC-2014-C230
October 2014

Analysis Components Investigation Report

This work was performed
Under contract to the

DEFENCE RESEARCH AND DEVELOPMENT CANADA - VALCARTIER

Contract Number: W7701-13-5551

Scientific Authority:
Alexandre Bergeron-Guyard

And prepared by
Fujitsu Consulting Canada Inc.
2000 Boulevard Lebourgneuf,
Bureau 300, Québec (Québec) G2K 0B8

Version 1.0

History

Version	Description	Author	Date
0.1	Table of Content	Sébastien Paquet	2013-06-12
0.2	First complete draft	Sébastien Paquet	2013-12-09
1.0	Final version taking into consideration comments from Luc Lamontagne, Hugo Larochelle and François Laviollette.	Sébastien Paquet	2013-12-20

Notes

The information contained herein is proprietary to Her Majesty and is provided to the recipient on the understanding that it will be used for information and evaluation purposes only. Any commercial use including use for manufacture is prohibited.

Table of Contents

Abstract

This report describes five analysis components for an Intelligence Virtual Analyst Capability (iVAC) that would help an iVAC to adapt and personalize how he searches for documents. These five components are: context and element of interest, document retrieval, solution feedback, context feedback and recommendation. The focus is on how the user's context could be used to improve the document retrieval task. We describe how the context could be implemented and how the system could use the user's feedback to improve the accuracy of the document retrieval task.

1 Introduction

1.1 Identification

This document is produced by Fujitsu Consulting (Canada) Inc. under contract W7701-135551/A, entitled “Intelligence Virtual Analyst Capability (iVAC) – Framework and Components”. It provides identification and description of 5 potential analysis components of the IVAC prototype.

1.2 Objectives

This report is produced in relation to an initiative at Defence R&D Canada (DRDC) Valcartier to develop an Intelligence Virtual Analyst Capability (iVAC) which could be defined as "a computerized software assistant supporting the intelligence analysts in sense making tasks, while being ultimately capable of taking on autonomous analytical tasks in concert with other analysts (virtual or human)".

The present report addresses a subset of IVAC capabilities. It focuses on the user’s context for a document retrieval task. The described analysis components would help an IVAC system taking the user’s context into consideration to adapt the document retrieval component in order to personalize and improve document’s search and discovery.

In this report, five analysis components are described:

1. **Context and Element of Interest.** How the context is structured and exploited to personalize search.
2. **Document Retrieval.** How documents are stored and which search engine is used.
3. **Solution Feedback.** How the system adapts the search using user’s feedbacks.
4. **Context Feedback.** How the system helps the user to specify his context.
5. **Recommendation.** How the system makes recommendations to the user.

For each component, we present the component’s requirements, propose possible approaches for accomplishing the requirements and explain how the component will be implemented in the prototype.

1.3 Overview

This document is organized as follows:

- Section 1 – Introduction: Identifies this document.
- Section 2 - Context and Element of Interest Component: Identifies requirements, proposes solutions and explains implementation for this component.

- Section 3 - Document Retrieval Component: Identifies requirements, proposes solutions and explains implementation for this component.
- Section 4 - Solution Feedback Component: Identifies requirements, proposes solutions and explains implementation for this component.
- Section 5 - Context Feedback Component: Identifies requirements, proposes solutions and explains implementation for this component.
- Section 6 - Recommendation Component: Identifies requirements, proposes solutions and explains implementation for this component.
- Section 7 - Conclusion: Provides the conclusion of the report and way ahead.
- Section 8 – Acronyms: Provides a list of acronyms used in this document.

2 Context and Element of Interest Component

2.1 Requirement

Based on the input from the current user, this analysis component must acquire information about the user context. In other words, the system must be aware of the user's context. Generally, the context refers to the interrelated conditions in which something exists or occurs. Within the iVAC, the context refers to any information that can be used to characterize the situation of the user. The notion of context applies both to the user itself, and to the analysis being performed. This is to say that a user can be performing an analysis task from his office in Ottawa (user context) and that this particular task may be focusing on retrieving documents about crop farmers in Afghanistan (analysis context).

The context model shall represent both the user and analysis contexts. This model must reflect elements such as: Mission Objective, User Role, Task, Location (where?), Identity (who?), Time (when?), Activity (what?) and Geopolitical/Cultural/Social/Economical elements. For example, a context may include:

- Mission objective: defend school rebuilding team
- User role: intelligence analyst
- Task: Analyze possible threats in the vicinity of the school.
- Location: A perimeter around the school.

The component (through a combination of Natural Language Processing (NLP) and Graphical User Interface (GUI)) must allow the user to specify and validate the current user and analysis contexts.

Moreover, for this component an Element of Interest (EOI) model must be defined. In a general sense an "element of interest" can be any element from a situation (person, vehicle, vessel, building...) that is deemed interesting based on a set of criteria.

The component must identify elements from a specified data source based on a set of criteria. For the current work effort, this component should leverage the legacy DRDC Valcartier software components and services which specialize in situation description.

The component (through a combination of NLP and GUI) must allow the user to identify the type of element which is of interest (as well as its location/source). It should also allow the user to identify and specify the criteria which are to be used to measure "interest". It should allow the user to visualize and provide feedback (validation and ranking) on the retrieved elements.

2.2 Proposed Approaches

The context could include the following information:

- Pre-defined fields;
- Free-form keyword description of context;
- Selected named entities from previous searches;
- Documents identified as relevant and irrelevant by the user.

The pre-defined fields could be used to specify common context information such as: mission objective, task, user role, etc. The values will be selected from pre-defined lists provided by the system. Additionally, the user could add keywords describing his current interest. The pre-defined fields and keywords provided by the user to define his context could be used as follows:

- When a document is identified as relevant by the user, the system will add the context information in the document's metadata.
- When executing a search, the system could add a criterion looking at the similarity between the user context and the documents metadata. This information could be added as an explicit search criterion and/or as a re-ranking criterion.

For the named entities, they could be proposed by the system to the user who will then select the relevant ones. The selected named-entities will then be added to the user context. The named-entities could be used by the user to specify his elements of interest.

The last part of the context is a set of documents labeled as relevant or irrelevant by the user. When the system displays a list of search results, the user can identify some of them as relevant and others as irrelevant. These labeled documents could then be used by the solution feedback component for relevance feedback, query expansion or document re-ranking.

2.3 Prototype Implementation

In the prototype developed for this contract, 3 out of the 4 components of the context will be implemented.

1. Pre-defined fields: Name, Mission, Function, Goal, Task and Operation.
2. List of keywords.
3. Labeled documents (relevant and irrelevant).

A GUI will be developed for the first two allowing the user to modify his pre-defined fields and keywords. For the labeled documents, a GUI will be provided for the user to identify which documents are relevant or irrelevant. These labels will be stored in the database and will thus be usable by the other analysis components as explained in the following sections.

When a document is identified as relevant by the user, the system will add the context information (i.e. pre-defined fields values and keywords) in the document's metadata. This metadata will be indexed by Solr and thus searchable.

The more direct use of the context will be used for query expansion where context search criteria will be added to the query. The query will thus have one part to search on the document's content and one part on the context metadata. Moreover, the user will have the possibility to select how much the context is considered compared to the keywords. Keywords will have a weight of *x* between 0 and 1 inclusively and the context will have a weight of 1-*x*. Figure 1 shows the relative importance of the keywords and the context from only considering the keywords to only considering the context. Therefore, the system will offer the full spectrum from a simple search (i.e. only considering keywords), to a contextualize search (i.e. considering keywords and context), to a context-based recommendation (i.e. only considering context).

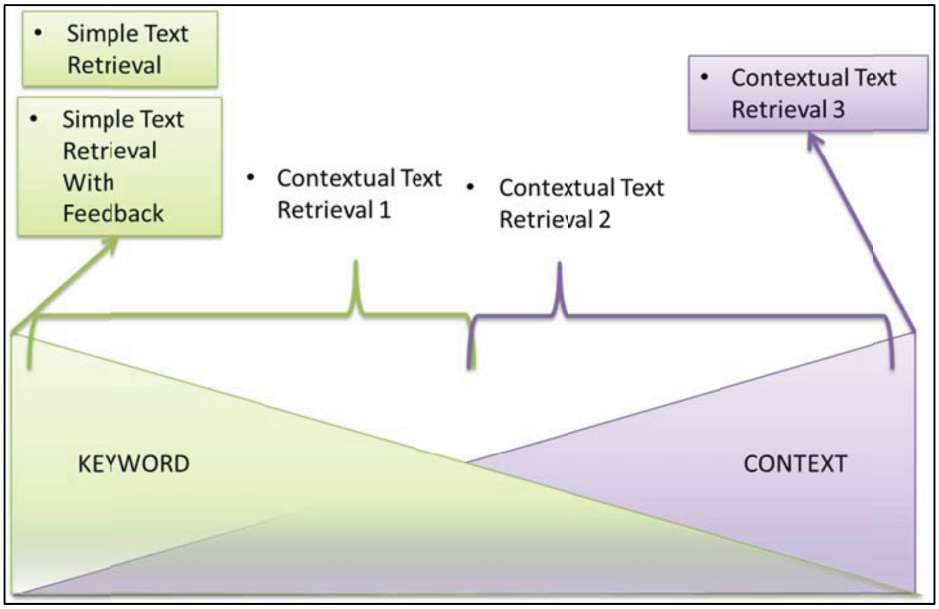

Simple Text Retrieval
Simple Text Retrieval With Feedback
Contextual Text Retrieval 1
Contextual Text Retrieval 2
Contextual Text Retrieval 3
KEYWORD
CONTEXT

Figure 1: Keyword/Context Relative Importance

3 Document Retrieval Component

3.1 Requirement

Based on a request (topic or keywords) from the user, and on the specifics of the context, this analysis component must retrieve relevant documents from a given document collection.

This component must leverage as much as possible the available DRDC Valcartier software components and services which specialize in text annotation and retrieval based on specific topics.

The component (through a combination of NLP and GUI) must allow the user to visualize and provide feedback (validation and ranking) on the retrieved documents.

3.2 Proposed Approaches

The ISTIP platform at DRDC already uses Alfresco as its document repository and offers the Document Repository service that allows managing documents in Alfresco, including search. Therefore, for iVAC, Alfresco will be used, but it will be upgraded in order to allow the addition of the Solr search engine that offers a more complete interface and provides functionalities required by other analysis components:

- Terms component: returns all the terms indexed;
- TermVector component: returns the term frequency, document frequency and Term Frequency–Inverse Document Frequency (TF-IDF[1]) values for any indexed document;
- The possibility to weight the importance of keywords in a query;
- The possibility to search on multiple document fields;
- The possibility to weight the importance of document fields in a query.

3.3 Prototype Implementation

For the prototype, it has been considered to upgrade Alfresco to version 4.2 because the new version provides connectors and configurations allowing the integration of Solr. However, the version of Solr that comes with Alfresco is an old version (1.4.1) that does not support all required functionalities. Since Alfresco modified Solr to integrate it in Alfresco, it is not possible to upgrade Solr.

Therefore, after discussion with DRDC, it has been decided to install Solr separately and feed the documents manually. This will enable the prototype to demonstrate the analysis components described in this document without the constraints of Alfresco. There will be no document management system; the documents will be stored on the file system.

[1] http://en.wikipedia.org/wiki/Tf%E2%80%93idf

The analysis components will communicate directly with Solr, using its REST interface in order to search for documents or retrieve information about indexed terms (using the Terms and TermVector components).

4 Solution Feedback Component

4.1 Requirement

For any given specific analysis task performed by the system, the user must be able to provide feedback to allow the analysis process to be validated and tuned. This feedback process will vary based on the analysis task performed. The validation may take different forms, for instance a ranking of results, or a selection of set of relevant results.

For the purpose of this contract the Solution Feedback Component must support the following analysis components: the document retrieval component and the element of Interest Identification Component.

The component (through a combination of NLP and GUI) must allow the user to provide feedback. Based on the feedback it must adjust its analysis process (algorithm) and provide new results. The system must validate the "updated" analysis parameters with the user (e.g.:"Now looking for individuals with weapon training AND a criminal record".) Based on this process, the system must gain a better understanding of how to perform a specific analysis task, and re-apply it in the future.

This component must leverage the work done by DRDC Valcartier on self-improving inference systems.

4.2 Proposed Approaches

Before presenting approaches for solution feedback, let’s describe the document retrieval process with feedback illustrated in Figure 2:

1. The user submits a request to the system. The request consists of keywords.
2. The system does pre-treatments on the request (e.g. stemming, stop words, normalisation, etc.).
3. The system searches in its document collection to find documents satisfying the request.
4. The system chooses some documents to show to the user. This choice could be made using a re-ranking function (described in Section 4.2.4) or all documents could be returned.
5. The system shows the documents to the user.
6. The user reads and evaluates some of the documents.
7. The user gives a feedback to the system by telling which documents are relevant and by proposing an order in which they should be presented.
8. The feedback component considers the user’s feedback and adjusts one of the system’s components in order to improve the system’s performances.

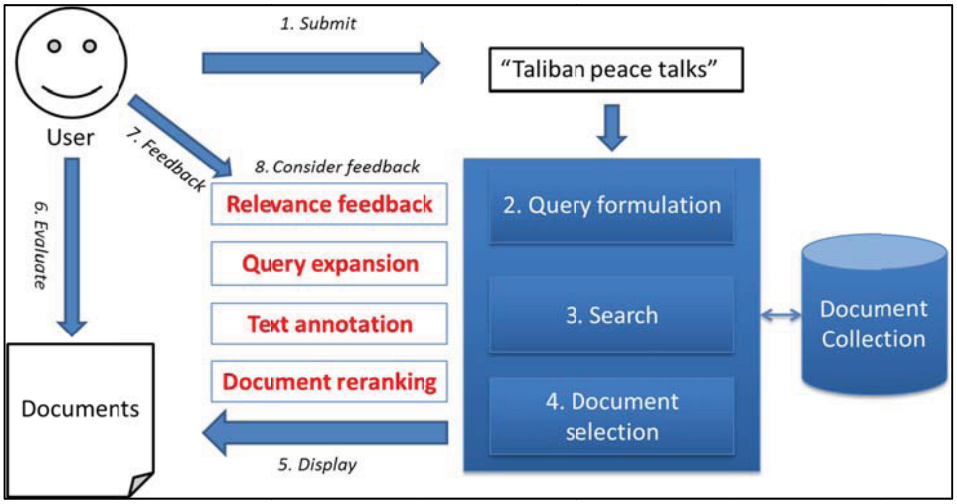


Figure 2: Document Retrieval with Feedback

The feedback component could use different approaches to improve the system's performances and in the following, we will present four types of approaches:

- Relevance feedback;
- Query expansion;
- Text annotation;
- Document re-ranking.

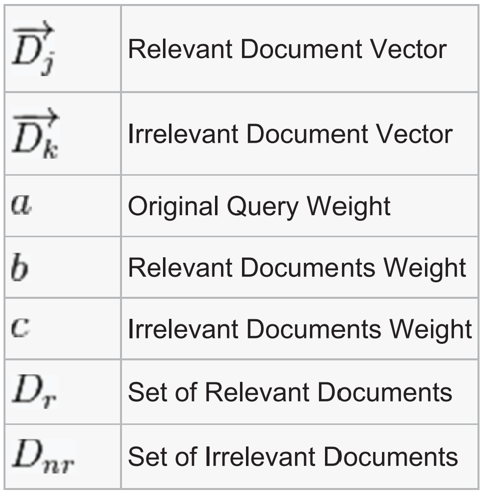
4.2.1 Relevance Feedback

Relevance feedback is an approach which modifies keywords weights in the request based on the relevant documents identified by the user. The main approach in this field is to use the Rocchio[2] equation to update the keywords weights:

$$\overrightarrow{Q_m} = (a \cdot \overrightarrow{Q_0}) + \left(b \cdot \frac{1}{|D_r|} \cdot \sum_{\overrightarrow{D_j} \epsilon D_r} \overrightarrow{D_j}\right) - \left(c \cdot \frac{1}{|D_{nr}|} \cdot \sum_{\overrightarrow{D_k} \epsilon D_{nr}} \overrightarrow{D_k}\right)$$

Variable	Value
$\overrightarrow{Q_m}$	Modified Query Vector
$\overrightarrow{Q_o}$	Original Query Vector

[2] See http://en.wikipedia.org/wiki/Rocchio_algorithm and http://nlp.stanford.edu/IR-book/html/htmledition/relevance-feedback-and-query-expansion-1.html



$\overrightarrow{D_j}$	Relevant Document Vector
$\overrightarrow{D_k}$	Irrelevant Document Vector
a	Original Query Weight
b	Relevant Documents Weight
c	Irrelevant Documents Weight
D_r	Set of Relevant Documents
D_{nr}	Set of Irrelevant Documents

In general terms, the Rocchio equation adds to the original weights the average importance of the word in the relevant documents and subtracts the average importance of the word in the irrelevant documents. The relative importance of these three terms is controlled with the three coefficients (*a*, *b* and *c*).

4.2.2 Query Expansion

Query Expansion[3] tries to improve search performances by adding words to the request. The main query expansion approaches are:

- use the most important words in the relevant documents;
- use words co-occurrences/collocations to add contextual words to the request;
- use a general linguistic resource like WordNet[4] (mostly the synonyms sets);
- use previous search logs.

4.2.2.1 Important Words

With this approach, the most important words in relevant documents are added to the query. There are different measures that could be used to determine which words are the most important ones.

To determine the importance of a word, we can use a measure such as TF*IDF[5] (Term Frequency * Inverse Document Frequency). TF*IDF is a numerical statistic which reflects how important a word is to a document in a collection of documents. For a document, it divides the term frequency (how many times the word appears in the document) by the document frequency (how many documents the word appears in).

[3] See http://dl.acm.org/citation.cfm?id=2071390 and http://nlp.stanford.edu/IR-book/html/htmledition/relevance-feedback-and-query-expansion-1.html

[4] http://wordnet.princeton.edu/

[5] http://en.wikipedia.org/wiki/Tf%E2%80%93idf

Moreover, the Rocchio equation presented previously can be used to add words to the query. It will add words that have an updated weight greater than zero. It could also be possible to use the TF*IDF measure for the term vectors used in the Rocchio equation.

Other approaches could also be used to identify important words, such as feature selection with Chi-squared[6] (taken from text classification) and use the centroids of relevant documents clusters.

4.2.2.2 Words Co-occurrences/Collocations

In a corpus, a collocation[7] is a sequence of words or terms whose co-occurences through the documents of the corpus are statistically higher than what is usually observed for a randomly picked sequence of words. Therefore, the approach for solution feedback is to identify the collocations of the query words in the relevant documents and to add the collocated words to the query.

The NLTK[8] library provides methods to identify collocations of words within a corpus of documents. There are other algorithms or software accessible on the web to identify collocations, but NLTK is already integrated in iVAC, it would thus be the easiest solution.

4.2.2.3 General Linguistic Resource

WordNet[9] is a lexical database of the English language where words (separated into the parts of speech: nouns, verbs, adjectives, and adverbs) are linked via semantic relationships. The smallest unit in a WordNet is a synset, which represents a specific meaning of a word. It includes the word, its explanation, and its synonyms.

One approach could be to add the direct synonyms to the query. Moreover, since the wordnet semantic relationships represent a graph, it is possible to calculate the similarity between two words, based on the distance between the two words in the graph[10]. Then, similar words and not just direct synonyms could be added to the query. The same approach could be used with an ontology.

4.2.2.4 Previous Search Logs

It is possible to use previous search logs to add relevant documents from similar searches to the query expansion analysis. This allows finding links between queries and words in documents and between queries. The central idea is to extract correlations between query terms and document terms by analyzing search logs. These correlations are then used to select high-quality expansion terms for new queries[11].

[6] http://nlp.stanford.edu/IR-book/html/htmledition/chi-square-feature-selection-1.html

[7] http://en.wikipedia.org/wiki/Collocation

[8] http://nltk.googlecode.com/svn/trunk/doc/howto/collocations.html

[9] http://shiffman.net/teaching/a2z/wordnet/

[10] http://www.codeproject.com/Articles/11835/WordNet-based-semantic-similarity-measurement

[11] http://research.microsoft.com/en-us/um/people/jrwen/jrwen_files/publications/QE-TKDE.pdf

4.2.3 Text Annotation

The idea is to annotate documents to identify named entities. For intelligence analysts, this normally means locations, people and organisations. By annotating the query and the documents, it would be easier to find words that have the same meaning.

The user can then have the possibility to give his feedback by validating the identified named entities and by selecting his named entities of interest. These named entities of interest could then be used for re-ranking and for query expansion.

Moreover, it could be useful to annotate documents using ontologies which would enable search on more general terms to find documents containing more specific words. For example, if an intelligence analyst searches for some terrorist group, it is possible that this way the system retrieves documents talking about members of this group even if the group itself is not mentioned in the documents.

4.2.4 Document Re-ranking

A machine learning approach could be used for document re-ranking. Indeed, a context, a query and a document can be converted in a feature vector and used by the system to learn to link this information with the notion of document relevance.

An example of feature vector could be:

1. Pre-defined field values in the user's context that are present in the document to order.
2. Keywords in the user's context that are present in the document to order.
3. Named entities in the user's context that are present in the document to order.
4. Words shared between the user's relevant documents and the document to order.
5. Words shared between the user's query and the document to order.
6. Similarity measure (e.g. TF-IDF) between the query and the document to order (useful to make sure the re-ranking is functional at the start when the model has been little trained).
7. Words shared between the query, the user's relevant documents and the document to order.

Features 1 to 4 are linked to the user's context, while Features 5 and 6 are linked to the query. Finally, Feature 7 adds interactions between the context and the query.

If we consider the re-ranking problem as a regression problem in some real interval, and where the outputs are then considered as relevance scores, many machine learning libraries could be used, namely some binary classification algorithms like the SVM, which outputs such a relevance score (that is called the margin). One interesting library is Vowpal Wabbit[12] (VW), developed by John Langford and his collaborators at Yahoo! and Microsoft. It provides many interesting features such as feature hashing, feature namespaces and topic modeling. One disadvantage of this library is that it only provides a command line interface, which may complicate its integration in DRDC's ITI.

[12] http://zinkov.com/posts/2013-08-13-vowpal-tutorial/

Another frequently used library is Scikit-learn[13], which is an open source library providing simple and efficient tools for data mining and data analysis.

Furthermore, these Machine Learning algorithms require a training set composed of relevant documents, tagged +1, and irrelevant documents, tagged -1. At the beginning, when no or few documents have been identified as relevant or irrelevant, it is possible to initialise the re-ranking model by generating fake data using some documents from the repository as the query and adding as relevant documents the query itself and as irrelevant documents a random subset of the other documents in the repository. In order to avoid selecting relevant documents, only documents that have a similarity measure with the query greater than a specified threshold will be considered as irrelevant documents. The approach would thus be:

- Choose a document randomly
- If Sim(query,doc) > threshold, then consider the doc as irrelevant document.

4.3 Prototype Implementation

4.3.1 Relevance Feedback and Query Expansion

For relevance feedback and query expansion, we will use the Rocchio equation as presented in Section 4.2.1. The user will make a first search and the system will present a first list of documents, then the user will identify documents as relevant or irrelevant and redo a search. The query will be adjusted using the user's feedback and the Rocchio equation, and then the system will present a new set of documents. The user will have the possibility to iterate using this process to refine his query.

The input parameters for the equation are:

- Set of documents identified as relevant by the user.
- Set of documents identified as irrelevant by the user.
- The user's search query. In the initial query, each term has a weight of 1. A Map class in Java should be used where the key would be the term and the value would be the weight.

For each relevant and non-relevant document, the system calls Solr to get the TF-IDF value for all terms in these documents. This will give one vector (use a Map in Java) for each document where the key is the term and the value is the TF-IDF value.

While constructing the vectors, the system adds each term in a list of terms, called FullTermsList. This list should contain all the terms present in all vectors, but only once. Therefore, all words should be distinct.

Afterwards, iterate over the FullTermsList, and for each term, calculate: $tw_m = (a \cdot tw_0) + \left(b \cdot \frac{1}{|D_r|} \cdot \sum_{\overrightarrow{D_j} \epsilon D_r} t_{TF-IDF}\right) - \left(c \cdot \frac{1}{|D_{nr}|} \cdot \sum_{\overrightarrow{D_k} \epsilon D_{nr}} t_{TF-IDF}\right)$

[13] http://scikit-learn.org

Variable	Value
tw_m	Modified Term Weight
tw_0	Original Term Weight
t_{TF-IDF}	The TF-IDF value of the term for a document.

- If a term is not present in a document (not in its vector), set its value to 0 for the calculation.
- If the calculated term weight is less than 0 do not add it to the resulting vector.

$\overrightarrow{Q_m}$ is the new query to send to Solr, containing all query terms with their modified weight. For example, the query could look like: q=terrorist^0.7 bomb^2.5.

4.3.2 Document Re-ranking

For the prototype, the Scikit-learn machine learning library will be used for document re-ranking. The Support Vector Machine (SVM) algorithm will be used to learn a predicting model that will then be used to give a relevance score to all documents returned by Solr and this score will be used to re-rank the documents.

The training phase could be executed at a specified interval (e.g. each week or each night) in order to rebuild the model with the new available information.

The training phase requires the following input:

- All documents tagged by any user as relevant or irrelevant, with their context metadata (it will be the training set).
- All the queries associated with these feedbacks.
- The context information of the users when they were making these queries.
- The vocabulary of the documents content (use the Solr Term component to retrieve the list of all terms used in the documents content, remove too frequent terms (e.g. present in more than 80% of the documents). There are also other approaches to filter the vocabulary (e.g. remove words present in a stop words list or remove words with low TF-IDF).
- The vocabulary of the context metadata (use the Solr Term component to retrieve the list of all terms used in the context metadata field and remove too frequent terms as for the content).

The feature vector will be of length equal to the length of the content vocabulary plus the length of the metadata vocabulary. For each term in the content vocabulary, there will be a 1 if the term is present in both the document and the query. For each term in the metadata vocabulary, there will be a 1 if the term is present in both the context information and the document's metadata. All other terms will have a value of 0.

Then, all feature vectors (one for each tagged document) and their relevance feedback will be passed to the SVM algorithm (Scikit-learn implementation) in order to learn a predictive model.

Afterwards, when a user makes a new query, the model is used to re-rank the documents. For each document returned by Solr, the system builds the feature vector using the document's content and metadata, and the user's query and context. The learned model is then used to give a relevance score to the document, allowing the system to re-rank the documents based on their score.

5 Context Feedback Component

5.1 Requirement

Context specification is addressed by the Context Specification Component. The nature of the context feedback component is akin to the solution feedback component. Indeed, the solution feedback component aims at improving and refining a solution to an analysis problem based on user feedback. The Context Feedback Component aims at refining the context based on the particular process that was applied to solving an analysis problem.

Here, context refers specifically to the user and analysis context, with a strong emphasis on the role, goal and task taken on by the user.

The objective is to compare the process being applied to a given analysis problem in a specific context, and evaluate its similarity to other, possibly similar problems in similar contexts.

> *Example: An intelligence analyst has a goal of identifying drug smugglers. The analyst asks for a list of all individuals with criminal records. In the past, a very similar analyst was working on a very similar goal. He had used a list of individuals with drug related convictions to reach a solution. Based on this similarity, the system also suggests a list of individuals with drug-related convictions to the analyst.*

This process entails the modeling of goals and tasks. It also involves the capability of measuring the similarity between these models.

This component must leverage the work done by DRDC Valcartier on self-improving inference systems and must be supported by a combination of NLP and GUI.

5.2 Proposed Approaches

As specified in Section 2.2, the context includes the following information:

- Pre-defined fields;
- Free-form keyword description of context;
- Selected named entities from previous searches;
- Documents identified as relevant and irrelevant by the user.

For the pre-defined fields, such as the user's role and task, the user has full control over them and can modify them whenever he wants. There is no suggestion by the system to improve this part of his context.

For the keywords, they are also provided by the user, but the system can also suggests to the user new keywords, based on similar contexts, in order to enrich the keywords list. All keyword's lists for all users are stored in the system, therefore the system can use similarity measures and algorithms such as k-nearest neighbors to identify similar list of keywords and identify important keywords missing from the user's keywords list.

The system can also propose the most important named entities found in relevant documents to the user. The user will then have the opportunity to validate the choices and add new named entities to his context. Moreover, an approach similar to the one proposed for the keywords could be used for the named entities.

Finally, for the relevant documents, it could be possible to use the approach presented for the recommendation system to identify documents that have been identified as relevant by other users having similar contexts. This would be done by doing searches that only consider the context information has search criteria. Another approach would be to identify other users that have identified the same relevant documents as the current user and to suggest to the current user all the other relevant documents identified by the other users.

To sum up, the context feedback component will help the user to improve his context by proposing new context elements based on the other users activities. This will enable collaborative improvement of each user context and thus collaborative improvement of each user information retrieval activities.

5.3 Prototype Implementation

No direct implementation of the Context Feedback component is planned for the prototype developed in this contract. However, there will be an indirect context refinement when the user identifies documents as relevant or irrelevant. Since these documents will be added to the user's context, they will refine his context. Moreover, the system will return relevant documents, which will help him refine his context.

6 Recommendation Component

6.1 Requirement

The iVAC could make recommendations on questions, requests or queries, and also on solutions based on other users' interactions with the system.

6.2 Proposed Approaches

In the context of the recommendation component, we suppose that the system is filtering the documents by using exclusively the other users' contexts. This problem is similar to collaborative filtering and also content filtering. These are five options:

1. **Find similar users.** The system needs to calculate the similarity between contexts (pre-defined fields). For this task, it is possible to simply calculate an exact match on the pre-defined fields (e.g. objective, task, role, etc.). However, it would be better to use a domain ontology that gives a taxonomy for these fields. This would allow the system to calculate the similarity based on the distance between the values in the ontology.
2. **Filter documents.** The system could show documents that have been identified as relevant by users having a similar context. This supposes that all users have the same documents collection. However, we consider that it would be better to construct a model of the relevant documents and rank the documents with this model. The model could be a combination of keywords and annotations.
3. **Build a pseudo-request.** Gather keywords used by users having similar context and use them as a query to find documents.
4. **Query using the context.** The system could recommend documents by executing queries exclusively using the user's context information, which will return documents identified as relevant by users having a similar context.
5. **Use the relevance feedback.** If document *x* has been identified as relevant by the user, the system could recommend other documents identified as relevant by users who have also identified *x* as relevant.

6.3 Prototype Implementation

For the prototype, option 4 will be implemented, which is to execute a search without keywords, considering only the user's context. This will return documents identified as relevant by users having similar context.

If there is enough time, option 5 could also be implemented. This would require the development of a GUI presenting the recommended documents for each document identified as relevant by the user.

7 Conclusion

In this report, we presented five analysis components that would enable an IVAC system to take the user's context into consideration in order to personalize and improve the document retrieval task.

We presented a flexible way to represent the context using pre-defined fields, keywords, entities and documents. It minimizes the user context maintenance actions, while enabling advanced analysis algorithms. In this report, this representation was then used for the solution feedback (i.e. relevance feedback, query expansion and document re-ranking), context feedback and recommendation components.

In each component's section, we presented how these components could be implemented in the IVAC prototype. The proposed implementation is a first step towards a more mature IVAC system. The objective was to describe solutions that would be implementable considering the available time left for the prototype development. Further work would be required to implement the other proposed approaches and also to fine-tune all the algorithms.

Finally, the current report only focussed on the document retrieval task which is a small portion of a complete IVAC system. Other research activities are required in advanced Intelligence Artificial fields in order to conceive and develop a complete virtual assistant capable of interacting with humans in their normal work environments.

8 Acronyms

Acronym	**Definition**
DRDC	Defence Research and Development Canada
EOI	Elements of Interest
GUI	Graphical User Interface
ISTIP	Intelligence Science and Technology Integration Platform
ITI	Information Technology Infrastructure
IVAC	Intelligent Virtual Assistant Capability
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
REST	Representational State Transfer
SVM	Support Vector Machine
TF-IDF	Term Frequency Inverse Document Frequency
VW	Vowpal Wabbit